Turtle Car Racing Game

1. Project Overview

Turtle Car Racing Game is a simple Python racing game built with the Turtle module.

The player controls a car using the keyboard while enemy cars move down the road. The player must avoid collisions and survive as long as possible.

The game introduces students to keyboard events, coordinates, loops, functions, conditions, random numbers, collision detection, and game animation.

2. Features

- Player car
- Enemy cars
- Keyboard controls
- Moving road
- Collision detection
- Score system
- Increasing difficulty
- Game over
- Restart option
- No external packages

3. Technologies

- Python 3
- Turtle
- Random
- Functions

- Loops
- Conditions
- Lists
- Keyboard events
- Coordinates

4. Folder Structure

```
turtle_car_racing/
□□□ main.py
□□□ game.py
□□□ README.md
```

5. How the Game Works

The player starts at the bottom of the road.

```
     ROAD
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□      □        □
□               □
□   □           □
□           □   □
□               □
□               □
□      □        □
□               □
□      PLAYER     □
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

The player moves left and right using the arrow keys.

Enemy cars continuously move toward the player.

If an enemy car touches the player, the game ends.

The longer the player survives, the higher the score.

6. Complete Game Code

game.py

```
import random
import turtle


class CarRacingGame:
    def __init__(self):
        self.screen = turtle.Screen()
        self.screen.title("Turtle Car Racing Game")
        self.screen.setup(width=600, height=700)
        self.screen.bgcolor("darkgreen")
        self.screen.tracer(0)

        self.score = 0
        self.game_over = False
        self.player_speed = 30
        self.enemy_speed = 8

        self.player = None
        self.enemies = []

        self.create_road()
        self.create_player()
        self.create_enemies()
        self.setup_controls()

    def create_road(self):
```

```
        road = turtle.Turtle()
        road.hideturtle()
        road.penup()
        road.color("gray")
        road.goto(-220, -350)
        road.begin_fill()

        for _ in range(2):
            road.forward(440)
            road.left(90)
            road.forward(700)
            road.left(90)

        road.end_fill()

        # Road center line
        line = turtle.Turtle()
        line.hideturtle()
        line.color("white")
        line.pensize(4)
        line.penup()

        for y in range(-330, 350, 70):
            line.goto(0, y)
            line.setheading(90)
            line.pendown()
            line.forward(35)
            line.penup()

    def create_player(self):
        self.player = turtle.Turtle()
        self.player.shape("square")
        self.player.shapesize(stretch_wid=1.5, stretch_len=1)
        self.player.color("blue")
```

```
        self.player.penup()
        self.player.goto(0, -280)

    def create_enemies(self):
        colors = ["red", "yellow", "orange", "purple"]

        for color in colors:
            enemy = turtle.Turtle()
            enemy.shape("square")
            enemy.shapesize(stretch_wid=1.5, stretch_len=1)
            enemy.color(color)
            enemy.penup()
            enemy.goto(
                random.randint(-180, 180),
                random.randint(100, 600)
            )
            self.enemies.append(enemy)

    def setup_controls(self):
        self.screen.listen()

        self.screen.onkeypress(self.move_left, "Left")
        self.screen.onkeypress(self.move_right, "Right")

    def move_left(self):
        if not self.game_over:
            x = self.player.xcor() - self.player_speed

            if x > -190:
                self.player.setx(x)

    def move_right(self):
        if not self.game_over:
            x = self.player.xcor() + self.player_speed
```

```
        if x < 190:
            self.player.setx(x)

    def move_enemies(self):
        for enemy in self.enemies:
            enemy.sety(enemy.ycor() - self.enemy_speed)

            if enemy.ycor() < -380:
                enemy.goto(
                    random.randint(-180, 180),
                    random.randint(350, 600)
                )

                self.score += 1

            if self.check_collision(self.player, enemy):
                self.end_game()
                return

        self.screen.update()

        if not self.game_over:
            self.screen.ontimer(self.move_enemies, 50)

    def check_collision(self, car1, car2):
        distance = car1.distance(car2)
        return distance < 35

    def end_game(self):
        self.game_over = True

        message = turtle.Turtle()
        message.hideturtle()
```

```
        message.color("white")
        message.penup()
        message.goto(0, 40)

        message.write(
            "GAME OVER",
            align="center",
            font=("Arial", 32, "bold")
        )

        message.goto(0, -20)
        message.write(
            f"Score: {self.score}",
            align="center",
            font=("Arial", 20, "normal")
        )

        message.goto(0, -70)
        message.write(
            "Close the window to exit",
            align="center",
            font=("Arial", 14, "normal")
        )

    def start(self):
        self.move_enemies()
        self.screen.mainloop()
```

7. Main Program

main.py

```
from game import CarRacingGame
```

```
def main():
    game = CarRacingGame()
    game.start()


if __name__ == "__main__":
    main()
```

8. How to Run

Open the project folder in VS Code.

Run:

```
python main.py
```

Use:

```
 Left Arrow
 Right Arrow
```

to control the player car.

Avoid the enemy cars and try to get the highest score.

9. What Students Learn

Turtle

Students learn how to create and control objects using Turtle.

```
car = turtle.Turtle()
car.shape("square")
```

`car.color("blue")`

Keyboard Events

```
screen.onkeypress(move_left, "Left")
screen.onkeypress(move_right, "Right")
```

These connect keyboard keys to functions. Turtle supports keyboard event bindings through `onkeypress()` and requires the screen to receive focus with `listen()`.

Coordinates

`player.goto(0, -280)`

Students learn that objects on the Turtle screen have `x` and `y` coordinates.

Random Numbers

`random.randint(-180, 180)`

This gives enemy cars different positions.

Collision Detection

`if car1.distance(car2) < 35:`

The game checks whether two cars are close enough to collide.

Game Loop

`self.screen.ontimer(self.move_enemies, 50)`

The enemy movement is repeatedly scheduled so the game continues running. Turtle provides `ontimer()` specifically for calling a function after a specified delay.

10. Basic Testing

Students should test:

- Left arrow moves the car left.
- Right arrow moves the car right.
- Player cannot leave the road.
- Enemy cars move downward.
- Enemy cars return to the top.
- Score increases when an enemy passes.
- Collision ends the game.
- Game-over message displays the score.

11. Student Challenges

Once the basic game works, give students these challenges.

Challenge 1

Add a **restart key**.

```
Press R to restart
```

Challenge 2

Increase the enemy speed after every 10 points.

```
Score 10 → Speed increases
Score 20 → Speed increases again
```

Challenge 3

Add three lives.

□ □ □

A collision removes one life instead of immediately ending the game.

Challenge 4

Add a **nitro/boost** key.

`SPACE → temporary speed boost`

Challenge 5

Let students customize:

- Car colors
- Road color
- Background
- Number of enemy cars
- Game title
- Speed
- Score system

12. Final Challenge

Ask each student:

> **"Apni racing game ka upgraded version banao."**

They must add at least **3 new features** of their own.

Possible ideas:

- Lives
- Nitro
- Levels
- Sound
- High score
- Different roads
- More enemy cars
- Fuel system
- Coins
- Speedometer

13. Project Concepts

By completing this project, students practice:

Variables

Functions

Conditions

Loops

Lists

Random Numbers

Coordinates

Keyboard Events

Collision Detection

Game Loop

14. How the Project is Organized

`main.py` starts the game.

`game.py` contains:

- Road
- Player
- Enemy cars
- Keyboard controls
- Movement
- Collision detection
- Score
- Game-over logic

This keeps the project organized while still being simple enough for a 10th class student to understand.

Visit haas.dev for more resources and guides.